SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I6P120

Volume 11 Issue 6, 182-187, June 2024

Original Article

Low Latent Fixed Width Multiplier for Error Resilient Computation

B.V. Srividya[1], S.P. Meharunnissa[2], Chetan Umadi[1], Nagarathna[1], Saravana Kumar[3]

[1]*Department of Electronics and Telecommunication Engineering, Dayananda Sagar College of Engineering, Karnataka, India.*
[2]*Department of Electronics and Instrumentation Communication Engineering, Dayananda Sagar College of Engineering, Karnataka, India.*
[3]*DFT Engineer, Radiant Semiconductors Pvt Ltd, Karnataka, India.*

[1]*Corresponding Author : srividyabv@gmail.com*

Received: 09 April 2024 Revised: 12 May 2024 Accepted: 09 June 2024 Published: 29 June 2024

Abstract - *Many applications in signal processing have an innate ability to tolerate a certain amount of computational mistakes. The human eye's limited capacity for perceiving images and videos makes approximation useful in computations. Hence, this concept of error resilience approach can be accommodated in the hardware to reduce the computational time in high-speed circuits. Basically, multiplication in the signal processing domain takes a longer time. Hence, approximate multipliers have been an area of interest in recent times. This paper initially deals with a detailed study of various approaches to approximate multipliers. Subsequently, a novel architecture for error-resilient multiplication is proposed wherein approximate partial products are obtained. The entire multiplication operation is divided into three modules. The architecture of these modules is designed such that it provides the approximate output. These three modules work in parallel, thereby increasing the throughput. Efficient components are used in the design to improve the performance. The proposed multiplier is designed and simulated using Cadence 45nm technology.*

Keywords - *Error resilience, Fixed width multipliers, Signal processing, Throughput, Low latency.*

1. Introduction

An approximate computation has become a viable method for designing digital systems with energy efficiency. Approximate computing approaches enable a significantly higher energy economy by removing the requirement for totally exact or completely predictable operation. Achieving energy minimization with the least amount of performance (speed) loss is greatly desired [1]. The computational core of digital signal processing in multimedia applications needs faster yet reliable arithmetic units, where multiplication has a greater share among all possible operations [2]. Hence, various methodologies in multiplier with a focus on performance metrics have the greater interest over the last two decades.

A large number of DSP cores are used to build algorithms for processing images and videos, with the end product being ready for human vision. The fact that the human eye has limited perceptual capabilities in observing an image or a video enables the use of approximation in computations by occasionally dropping a few of the frames. Apart from image and video processing applications, there exist additional domains in which the precision of arithmetic operations is not required for the functioning of the system [4, 5]. Specifically, performance metrics-oriented application domains share an intrinsic tolerance for small and negligible errors [3]. The foundation of approximate computing is the discovery that, in certain situations, allowing for bounded approximations can result in a disproportionate gain in performance and energy while maintaining acceptable result accuracy, even when executing exact calculations costs a large amount of resources. Consider two distinct classes that yield comparable classification results in a set of sample items as an additional data analysis example. It is exceedingly challenging, if not impossible, to determine which is superior for classifying newly discovered items.

Such approximates may be added as arbitrary circuits in the Boolean/High-level descriptions, or they may be incorporated into the main building blocks that are utilized in the circuits. The goal of approximate arithmetic is to create simple arithmetic operations, like multipliers and adders that can be used in programmable computers to supplement accurate arithmetic operations [2]. The idea that these arithmetic units perform makes sense and provides fast results compared to exact computation data paths. In the context of Very Large-Scale Integration (VLSI), leakage power refers to the power that a digital circuit uses even while it is not actively operating. The power lost during the charging and discharging

of the load capacitance at the cell's output is known as the switching power of a driving cell. The low latency means the six least significant bits are made zero. The least significant six bits of each partial product are hardwired to zero during the computation of the final product from the partial products, ensuring speed improvement and delay reduction.

2. Literature Review

In order to achieve superior design metrics, Gupta et al. [2] suggested a number of approximate adder designs that eliminated some part of the logic that was included in conventional adders. This model results in shorter critical path designs, enabling voltage scaling possibilities. Also, mathematical models are derived for power consumption and error calculations in approximate adders. Since the main element in the multiplier is adders in the subsequent stage, the analysis of inaccurate adders is the greater interest of concern when dealing with performance analysis of approximate multipliers. A simulation result puts a noteworthy result of up to 69% power saving as compared to accurate adders.

In order to increase accuracy, Z Babic et al. [4] suggested a log-based pipelined approximation using an iterative process. The iterative MA multiplier is proposed to be performed in parallel using a single correction circuit. The results, when implemented on the Xilinx xc3s500e FPGA, reveal that power consumption increases only a little, from 2% (one correction term) to 16% (three correction terms). Along with this, the maximum computational delay rises by 30% to 45% for every additional correction circuit.

An approximate multiplier and adder based on the broken array multiplier approach was suggested by H. R. Mahdiani et al. [6]. The suggested paradigm offers faster, more affordable, and more effective implementations. The efficiency with which the suggested BAM builds a three-layer Neural Network (NN) for face recognition and a defuzzification block, which is utilized in a fuzzy inference engine, is demonstrated by the results of simulation and synthesis. Here, an array multiplier and a ripple carry adder are used to build the precise model, whereas a single multiplier and an adder comprise the data path and the critical path. The synthesized results from the Leonardo Spectrum tool show that the area delay product, in comparison with the precise model for 0.13µm standard cell library CMOS technology, clearly suggests greater improvements.

F. Farshchi et al. [7] apply the above-proposed BAM to booth multiplier, a modified arrangement which helps to deal with signed binary computations. The system's power consumption was reduced by almost 50% due to the suggested approximation blocks, which also resulted in a 6dB peak reduction in the signal-to-noise ratio. Additionally, the enhanced power-delay product outperforms traditional adders by roughly 65%. To determine the suggested model's power consumption, the design is synthesized in a standard cell of 90nm CMOS technology using the synopsys design compiler.

In order to shorten the critical path, K. Bhardwaj et al. [13] proposed an Approximation Wallace Tree Multiplier (AWTM) with a carry-in prediction. In comparison to the case of employing an accurate Wallace Tree Multiplier, AWTM was employed in this work's real-time benchmark image applications, demonstrating reductions in power and area of roughly 40% and 30%, respectively, without sacrificing image quality. Synthesis results from Cadence RTL provide power, and area requirements are compared with accurate Wallace tree arrangement. Also, accuracy and acceptance probability for a 16*16 multiplication are generated for 5000 random combinations, and various accuracy design metrics are tabulated.

The rest of the brief is organized as follows. Section 3 brings the background about signed multipliers and the approximation involved in that, Section 4 briefs about a proposal for approximate multiplier architecture with expected results, and Section 5 concludes this article.

3. Background

3.1. Two's Complement Multiplication

To illustrate with an example, let us consider that A and B are two numbers that are represented in 2's complement format. The input A has m bits while the input B has n bits.

$$A = -A_{m-1}2^{m-1} + \sum_{i=0}^{i=m-2} A_i 2^i \quad (1)$$

$$B = -B_{n-1}2^{n-1} + \sum_{j=0}^{j=n-2} B_i 2^i \quad (2)$$

Then, the product P, which has m+n bits, can be written as,

$$P = A_{m-1}B_{n-1}\ 2^{m+n-2} + \sum_{i=0}^{m-2}\sum_{j=0}^{n-2} A_i\ B_j\ 2^{i+j} - \sum_{i=0}^{i=m-2} A_i\ \ B_{n-1}\ 2^{n-1+i} - \sum_{j=0}^{j=n-2} A_{m-1}\ B_j\ 2^{m-1+j} \quad (3)$$

The two subtractions in Equation (3) can be expressed as an addition of 2's complement numbers; thereby, the above equation can be realized with the help of all adders instead of subtractions.

As in DSP processing, the multiplication process gets a greater share of up to 80% of computational capabilities, and plenty of architectures are devised to improvise the implementation features of Equation (3).

To understand the fixed-width multiplication, consider A and B as 8 bit wide. Hence, after multiplication, the product obtained will be 16 bits. In fixed width multiplier, only the upper 8 bit output is considered while truncating the lower 8 bits, which is shown in Figure 1.

As the least bits have much lower significance as compared with higher bits, truncating the LSB would result in a greater reduction in both hardware utilization and

computational time. The product P for an 8 * 8 signed fixed multiplier is given as:

$$P = MSB + LSB$$

$$\Longrightarrow P = \sum_{i=n}^{i=2n-1} P_i 2^i \quad (4)$$

As the LSB part of the product is truncated to hardwired '0'.

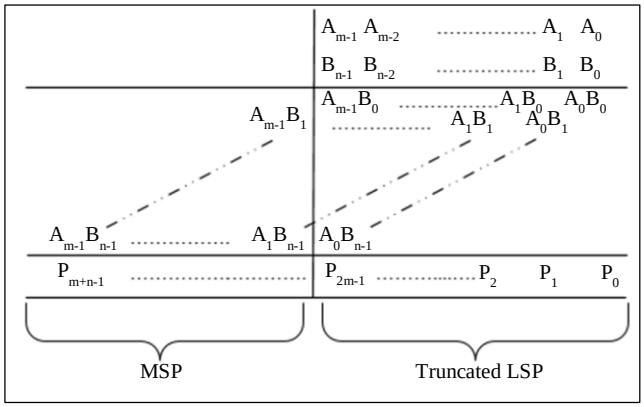

Fig. 1 Representation of partial products

4. Proposed Fixed Multiplier

The proposed architecture for an 8*8 fixed multiplier is detailed in Figure 2. The partial products obtained are separated as MSP and LSP. Instead of truncating the entire LSB part as defined in the previous section, the least six bits are hard-wired to '0', and the bits P6 and P7 are used to round off the LSB of the multiplier output. This is done to improve the result close toward exact.

Again, the computational capabilities of MSP are greatly improvised by subdividing into three major parts, which are shown in Figure 3. The three blocks work independently, and partial outputs are obtained.

								A7B0	A6B0
							A7B1	A6B1	A5B1
						A7B2	A6B2	A5B2	A4B2
					A7B3	A6B3	A5B3	A4B3	A3B3
				A7B4	A6B4	A5B4	A4B4	A3B4	A2B4
			A7B5	A6B5	A5B5	A4B5	A3B5	A2B5	A1B5
		A7B6	A6B6	A5B6	A4B6	A3B6	A2B6	A1B6	A0B6
	A7B7	A6B7	A5B7	A4B7	A3B7	A2B7	A1B7	A0B7	
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6

Fig. 2 Partial product array of 8 with 2 bits for improving accuracy

The partial results are combined with the suitable logic circuit to get the final product. As the Multiplication is carried out in the independent flow of signals, the latency of this proposed arrangement will be much less on par with the existing architectures. Realization of these three major parts can be done with an efficient combinatorial circuit, which aids in speeding up the results towards low latency.

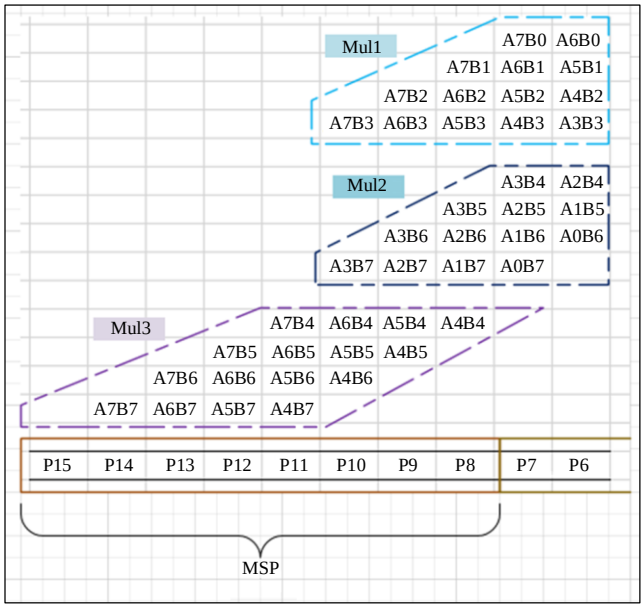

Fig. 3 Multiplication modules for computing MSP

The complete architecture of the proposed 8*8 fixed-width multiplier with three major parts is shown in Figure 4. The inputs A and B are appropriately routed to these modules, and the partial products of these modules can be added using a customized adder designed to get the final results in very little latency. Also, the least 6 bits of the product (P_{5-0}) are hardwired to logic '0', and P_6 and P_7 are used to make the multiplier result close to the exact output.

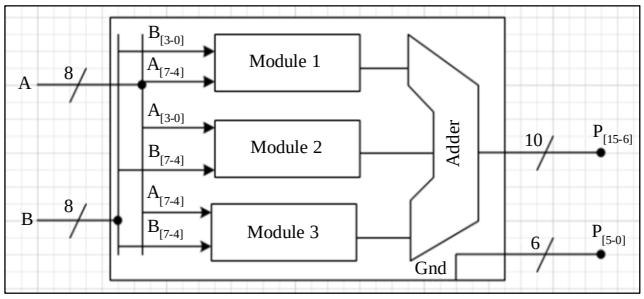

Fig. 4 Realization of modified fixed-point multiplier

The major sub-parts are realized with the help of combinatorial logic, much similar to that of the circuit shown in Figure 5. In fact, since the architecture deals with approximate multipliers, the full adders can be replaced with various approximate adders, which consume less space and operate faster. Replicated tree arrangement of such AND

Adder logic will be used in realizing each module as shown in Figure 6.

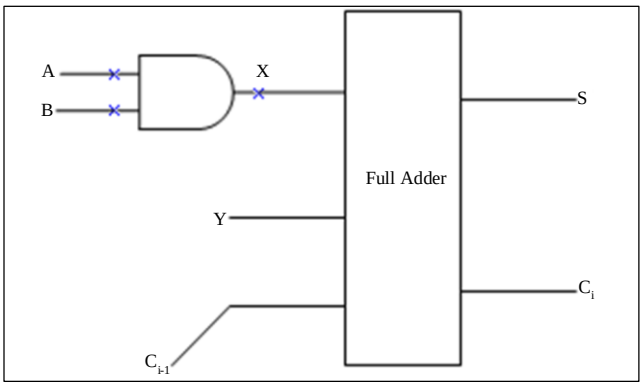

Fig. 5 AND logic with adder logic (ANADD logic)

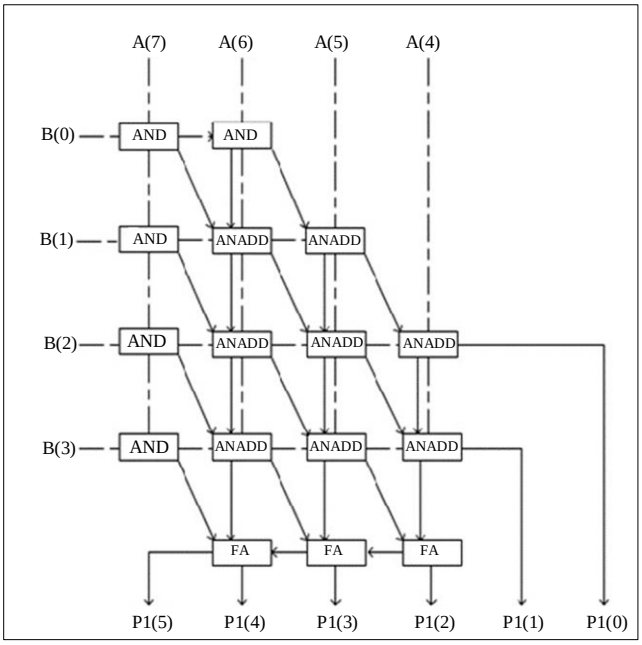

Fig. 6 Module 1 realization using ANADD logic

The 6 bit partial product ($P1_{[5-0]}$) of module 1 can be obtained with a propagation delay of 6 Adder delays; similarly, 6 bit partial product ($P2_{[5-0]}$) would have been available after the same delay, as each module works independently. After 3 adder delays, the least 2 – bits of modules 1 and 2 are available, with which the carry is generated to the final adder. Since module 3 needs 8 adder delay to generate 8 bit partial outputs ($P3_{[7-0]}$), it will not have any impact on the overall delay since least two partial products of Module 3 are available after one adder delay itself.

Finally, an adder is realized to perform the addition of these 3 partial products. Figure 7 proposes one such arrangement. The generated carry from the least 2 bits of modules 1 and 2 is routed to a synchronous carry save adder to generate the MSB part of the result.

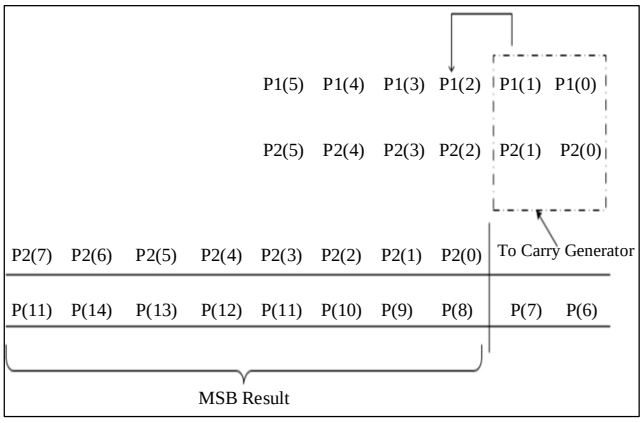

Fig. 7 Final adder realization

5. Implementation

The proposed multiplier, as shown in Figure 4, has been coded using verilog and synthesized using genus in cadence. The script file written using transaction control language with 45nm technology provides a detailed report on the area, power, and timing summary for the proposed multiplier. The synthesized RTL is shown in Figure 8.

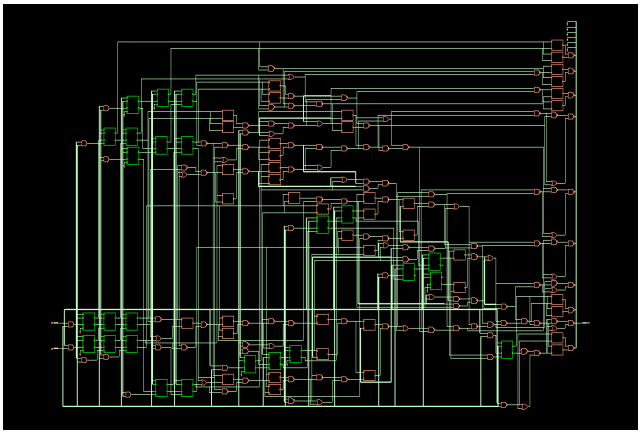

Fig. 8 RTL implementation of the proposed multiplier

Figure 9 depicts the pictorial representation of the simulation results for the error resilient multiplier, where the 6 least significant bits are forced to logic zero.

To obtain the transient power of the proposed multiplier the following circuits have been implemented using GPDK 45nm technology. The circuit includes Binary AND, OR, INVETER, NAND, NOR, and XOR. Combinational circuits such as half adder, full adder, parallel Adder, the proposed ANADD component and the 8 bit proposed multiplier have been implemented using cadence virtuoso.

The logical diagram shown in Figure 10 is the proposed error-resilient multiplier. The simulation results obtained on spectre have been functionally verified for the correctness and working of the multiplier. Subsequently, DC power analysis

and transient power analysis have been carried out to determine the DC power and the transient power, which is shown in Figure 11. The transient power obtained is in the range of 44.583uW to 44.587uW. The DC power is in the range of 60uW to 180uW.

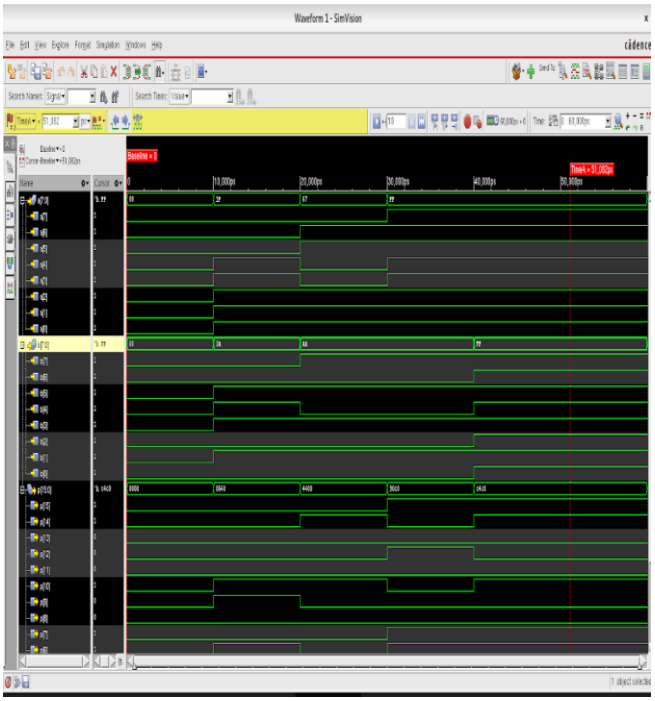

Fig. 9 Functional verification of proposed multiplier

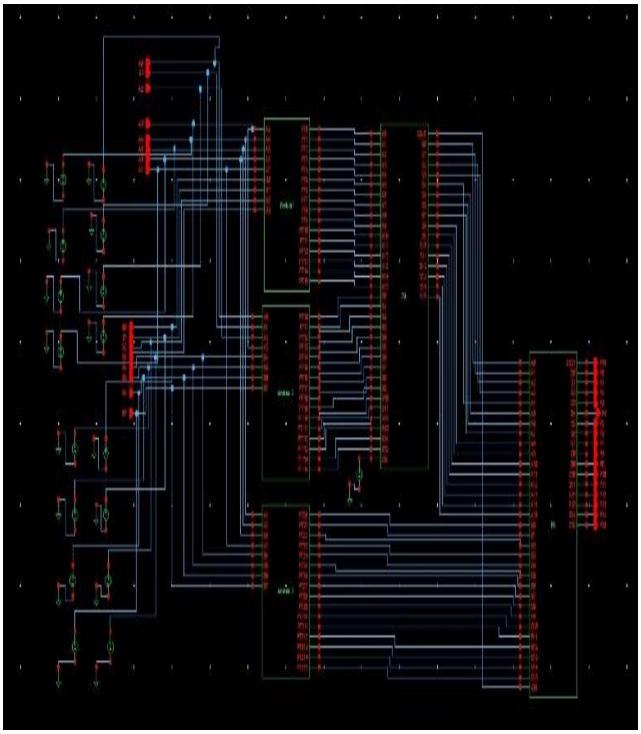

Fig. 10 Logical diagram of the proposed multiplier

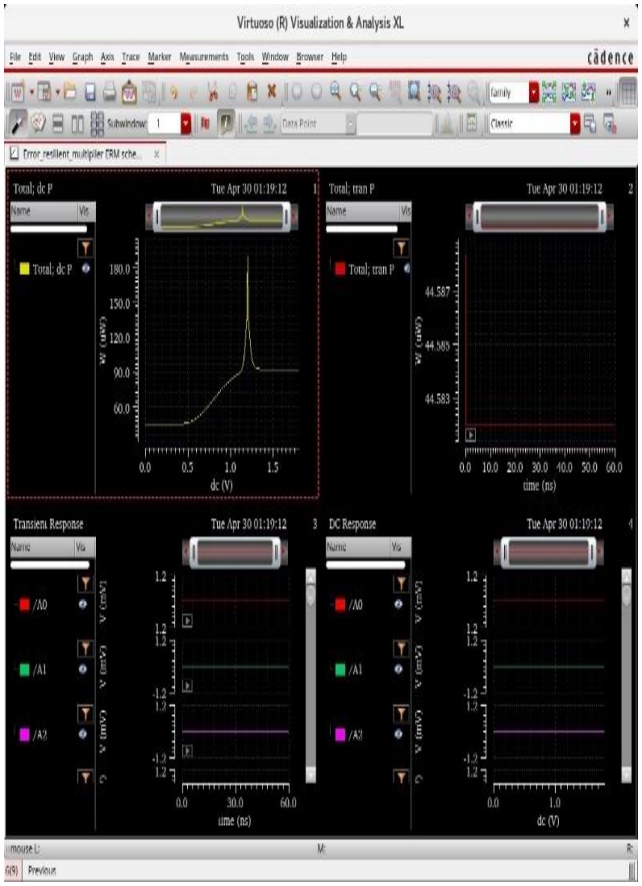

Fig. 11 Power analysis of the proposed multiplier

6. Performance Evaluation

The proposed fixed width multiplier is designed and coded in Verilog and subsequently implemented using Cadence 45nm technology. The simulation results have functionally verified the multiplication operation.

The leakage power obtained is 0.979 nW, and the internal switching power is 0.617 uW. Also, the transient power obtained is in the range of 44.583uW to 44.587uW. The DC power is in the range of 60uW to 180uW. Area occupancy reports 385 cells and 8963.125 as cell area.

7. Conclusion

The fixed-width high-speed approximate multiplier can be designed by dividing the partial products into three modules. Parallel computation improves the speed. For each module, a customized data path is developed, and independency is an added advantage to the proposed architecture.

Since approximate multipliers are the key to image processing, various means of constraint improvements can be achieved by adapting different strategies. The proposed multiplier utilizes fewer gates and combinational circuits to yield the result as compared to the conventional multiplier.

References

[1] Massimo Alioto, "Ultra-Low Power VLSI Circuit Design Demystified and Explained: A Tutorial," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 1, pp. 3-29, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[2] Vaibhav Gupta et al., "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124-137, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jie Han, and Michael Orshansky, "Approximate Computing: An Emerging Paradigm for Energy-Efficient Design," *2013 18th IEEE European Test Symposium (ETS)*, Avignon, France, pp. 1-6, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[4] Z. Babić, A. Avramović, and P. Bulić, "An Iterative Logarithmic Multiplier," *Microprocessors and Microsystems*, vol. 35, no. 1, pp. 23-33, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[5] Rangharajan Venkatesan et al., "MACACO: Modeling and Analysis of Circuits for Approximate Computing," *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, USA, pp. 667-673, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[6] H.R. Mahdiani et al., "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850-862, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[7] Farzad Farshchi, Muhammad Saeed Abrishami, and Sied Mehdi Fakhraie, "New Approximate Multiplier for Low Power Digital Signal Processing," *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, Tehran, Iran, pp. 25-30, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[8] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," *2011 24th Internatioal Conference on VLSI Design*, Chennai, India, pp. 346-351, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[9] D.R. Kelly, B.J. Phillips, and S. Al-Sarawi, "Approximate Signed Binary Integer Multipliers for Arithmetic Data Value Speculation," *Proceedings of the 2009 Conference on Design & Architectures For Signal And Image Processing*, pp. 97-104, 2009. [Google Scholar] [Publisher Link]

[10] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo, "Low-Power High-Speed Multiplier for Error-Tolerant Application," *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, Hong Kong, China, pp. 1-4, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[11] Amir Momeni et al., "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984-994, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] Kartikeya Bhardwaj, and Pravin S. Mane, "ACMA: Accuracy-Configurable Multiplier Architecture for Error-Resilient System-on-Chip," *2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, Darmstadt, Germany, pp. 1-6, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[13] Kartikeya Bhardwaj, Pravin S. Mane, and Jörg Henkel, "Power and Area-Efficient Approximate Wallace Tree Multiplier for Error-Resilient Systems," *Fifteenth International Symposium on Quality Electronic Design*, Santa Clara, USA, pp. 263-269, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[14] John N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512-517, 1962. [CrossRef] [Google Scholar] [Publisher Link]

[15] V. Mahalingam, and N. Ranganathan, "Improving Accuracy in Mitchell's Logarithmic Multiplication Using Operand Decomposition," *IEEE Transactions on Computers*, vol. 55, no. 12, pp. 1523-1535, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[16] Srinivasan Narayanamoorthy et al., "Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, pp. 1180-1184, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[17] Soheil Hashemi, R. Iris Bahar, and Sherief Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, USA, 2015, pp. 418-425, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[18] Chia-Hao Lin, and Ing-Chao Lin, "High Accuracy Approximate Multiplier with Error Correction," *2013 IEEE 31st International Conference on Computer Design (ICCD)*, Asheville, USA, pp. 33-38, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[19] Cong Liu, Jie Han, and Fabrizio Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, pp. 1-4, 2014. [CrossRef] [Google Scholar] [Publisher Link]